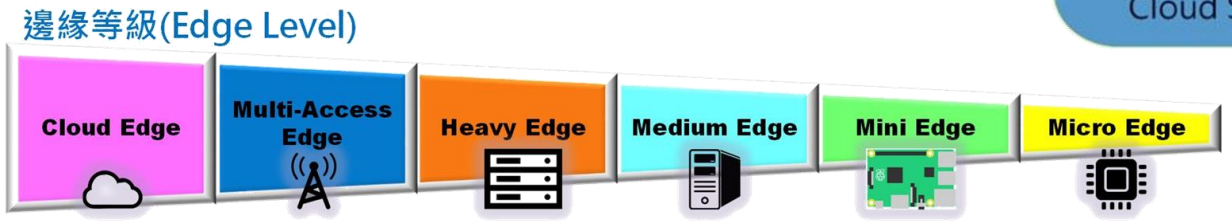
國立臺灣科技大學
NATIONAL TAIWAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Computer Science & Information Engineering
資 訊 工 程 系

人工智慧與邊緣運算實務

7.2 邊緣智慧案例實作【物件偵測】



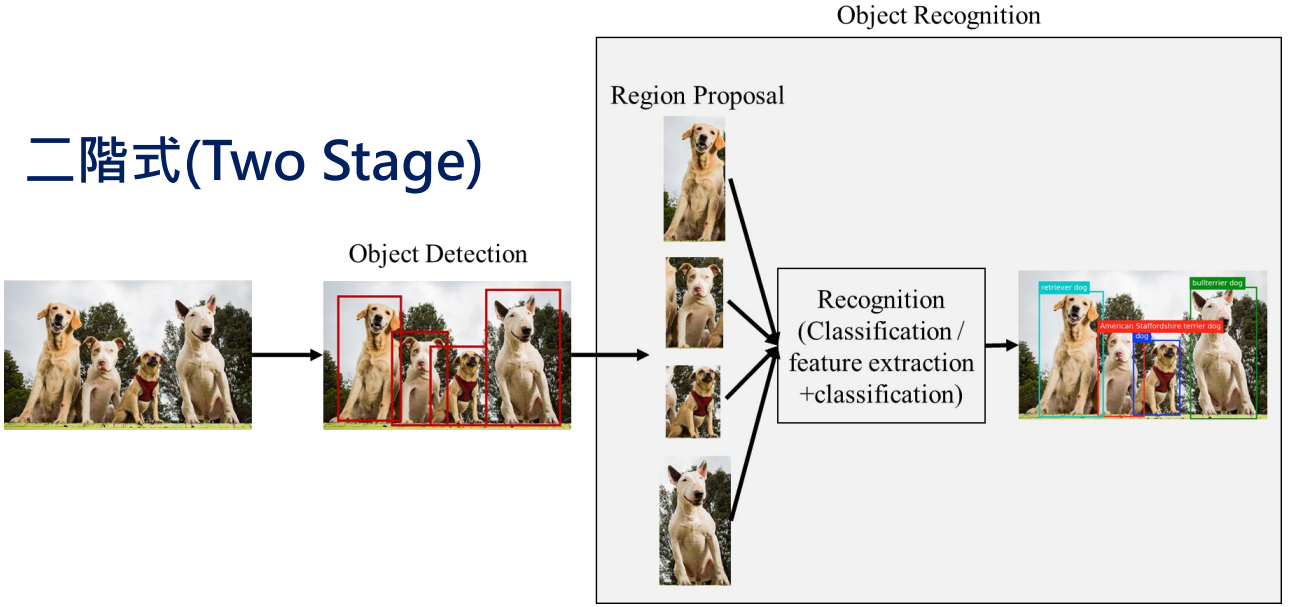
資訊工程系　許哲豪 助理教授

7.2 物件偵測

- 物件偵測簡介
- 預訓練模型推論
- 自定義模型訓練及推論

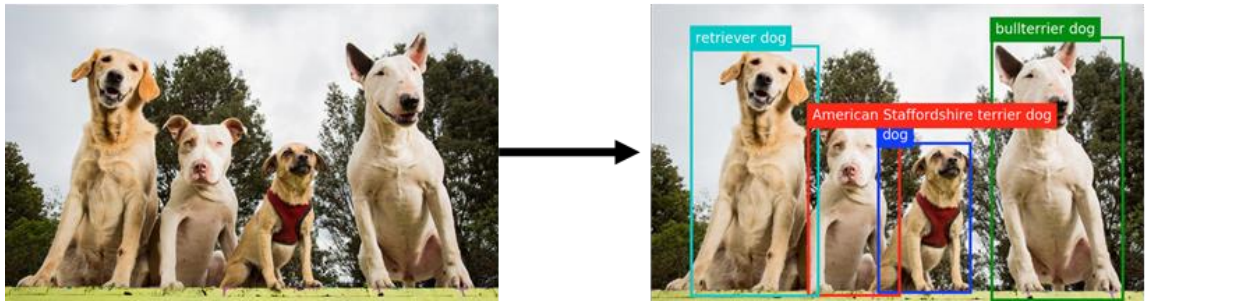
常見物件偵測模型

二階式(Two Stage)



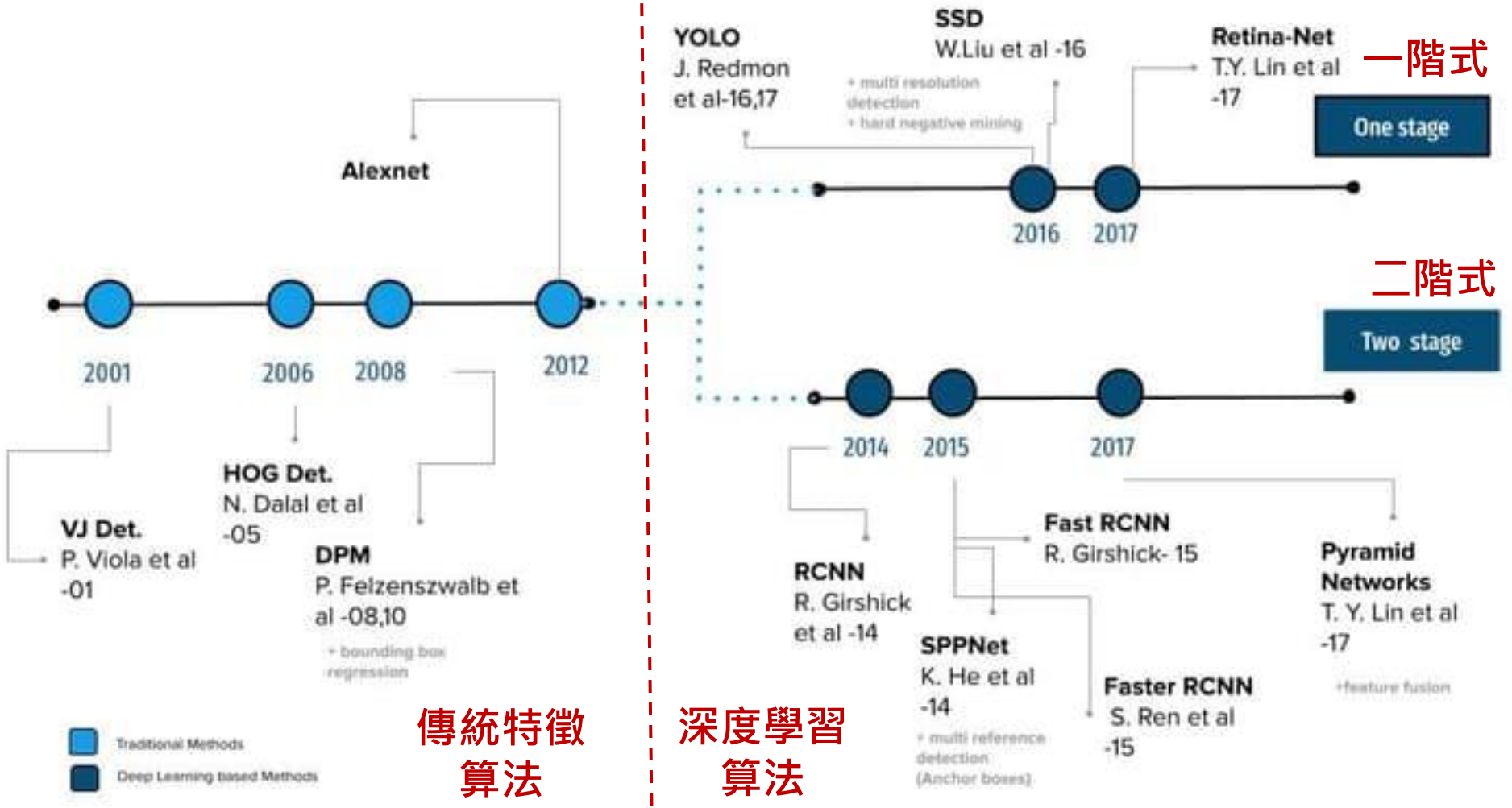
先找出物件可能區域(Region Proposals)再進行物件辨識，如
R-CNN
Fast R-CNN
Faster R-CNN
…

一階式(One Stage)



物件偵測及辨識一起完成，如
SSD
YOLO
…

物件偵測技術演進



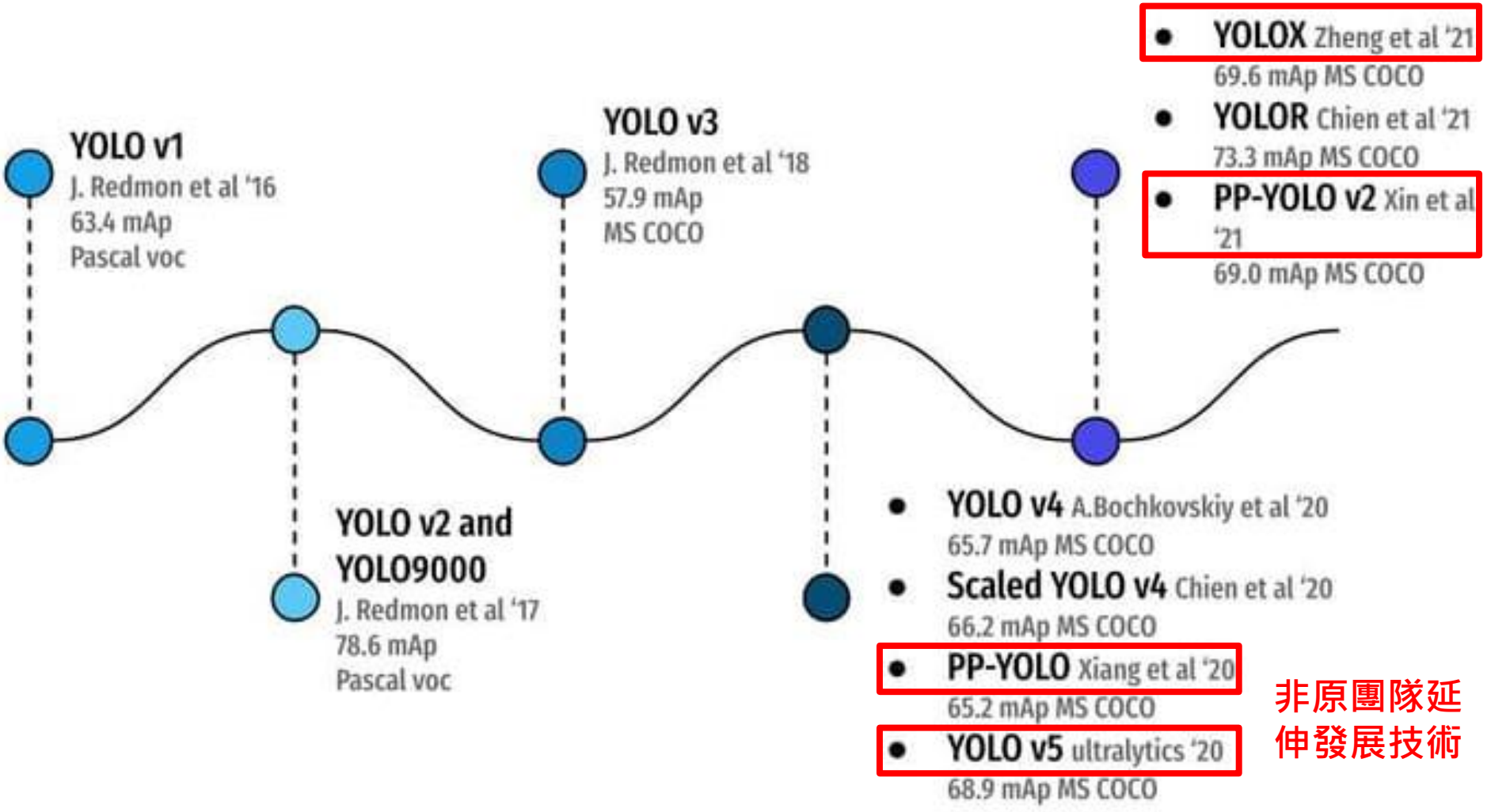
資料來源：https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/

物件偵測YOLO & Darknet

YOLO (You Only Look Once)

YOLO : https://pjreddie.com/darknet/yolo/
DarkNet : https://github.com/AlexeyAB/darknet

Yolo家族演進



資料來源：https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/

YOLOv4支援框架及工具

- Pytorch
- TensorFlow
- OpenCV
- **Intel OpenVINO**
- ONNX
- TensorRT+tkDNN
- DeepStream
- Triton Inference
- DirectML
- OpenCL
- HIP
- ROS
- Xilinx Zynq Ultrascale
- Amazon Neurochip
- TVM
- Netron

OpenVINO支援Yolo家族清單

- Public Pre-Trained Models

2021.4

- yolo-v1-tiny-tf
- yolo-v2-tf
- yolo-v2-tiny-tf
- yolo-v3-tf
- yolo-v3-tiny-tf
- yolo-v4-tf
- yolo-v4-tiny-tf

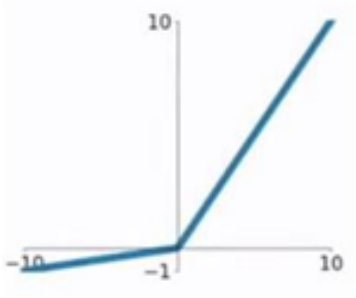
2022.1

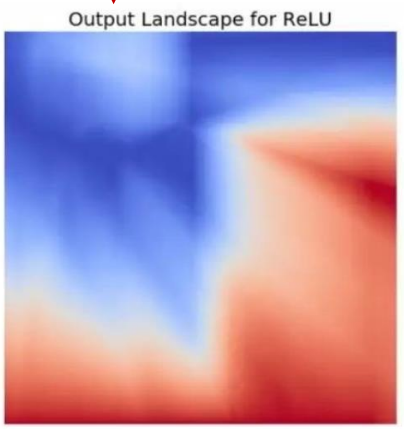
- yolo-v3-onnx
- yolo-v3-tiny-onnx
- yolof
- yolox-tiny

YOLOv4 Model Zoo

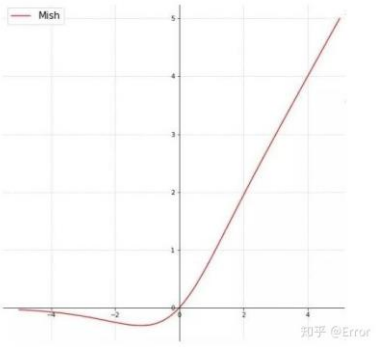
Model	Backbone	Neck	Plugin	V100 FPS	BFLOPs
YOLOv4	CSPDarknet53 with Mish Activation	PANet with Leaky activation	SPP	62@608x608, 83@512x512	128.5@608x608
YOLOv4-Leaky	CSPDarknet53 with Leaky activation	PANet with Leaky activation	SPP		128.5@608x608
YOLOv4-SAM-Leaky	CSPDarknet53 with Leaky activation	PANet with Leaky activation	SPP, SAM		130.7@608x608
YOLOv4-Mish	CSPDarknet53 with Mish activation	PANet with Mish activation	SPP		128.5@608x608
YOLOv4-SAM-Mish	CSPDarknet53 with Mish activation	PANet with Mish activation	SPP, SAM	61@608x608, 81@512x512	130.7@608x608
YOLOv4-CSP	CSPDarknet53 with Mish activation	CSPPANet with Mish activation	SPP		
YOLOv4-CSP-SAM	CSPDarknet53 with Mish activation	CSPPANet with Mish activation	SPP, SAM		

Leaky ReLU



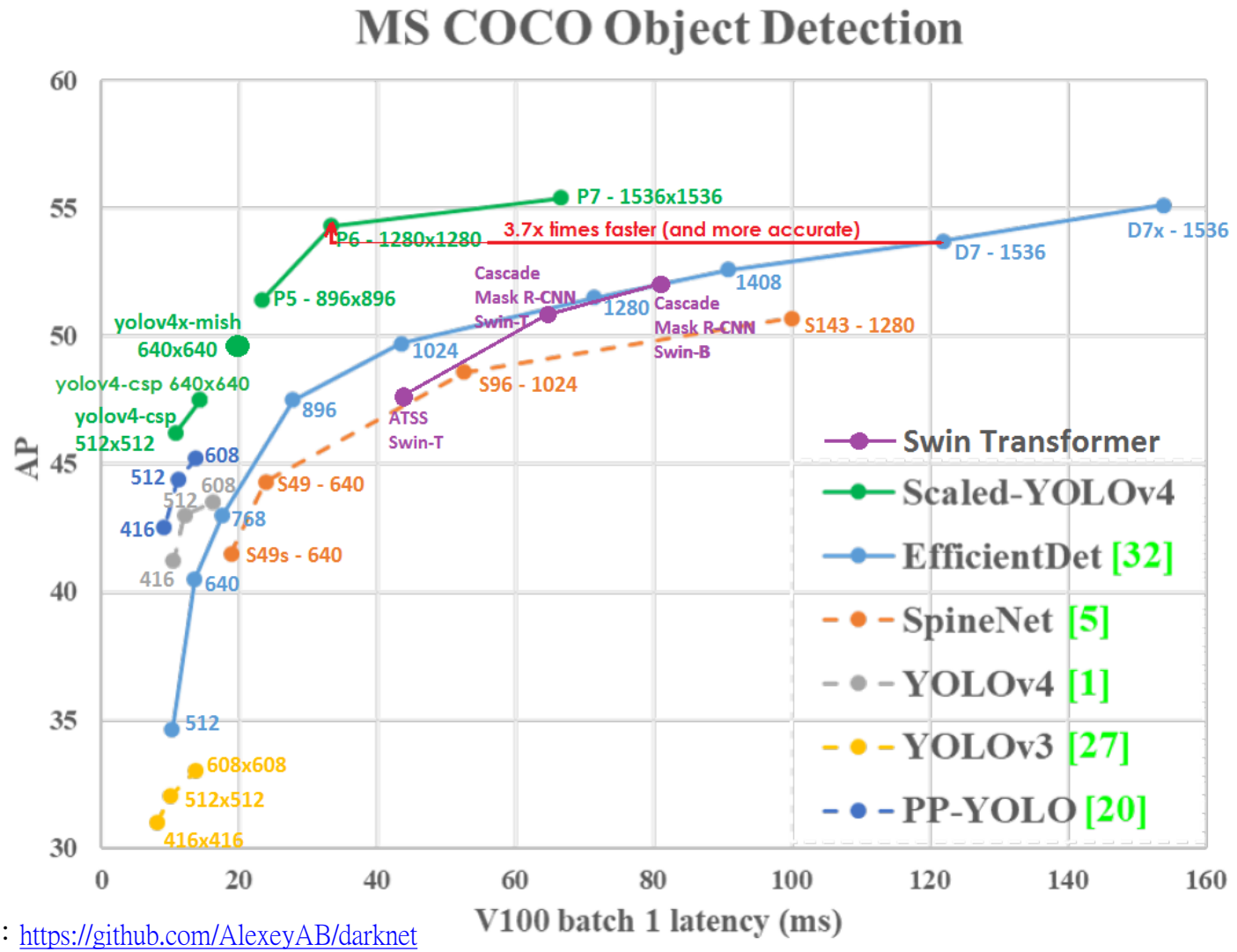


Mish=x * tanh(ln(1+e^x))



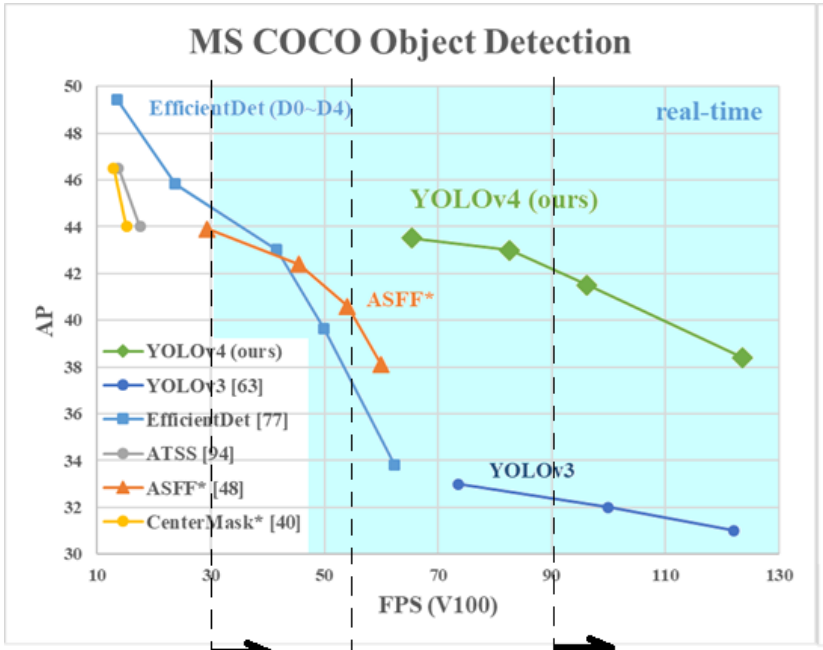
資料來源：https://github.com/AlexeyAB/darknet/wiki/YOLOv4-model-zoo

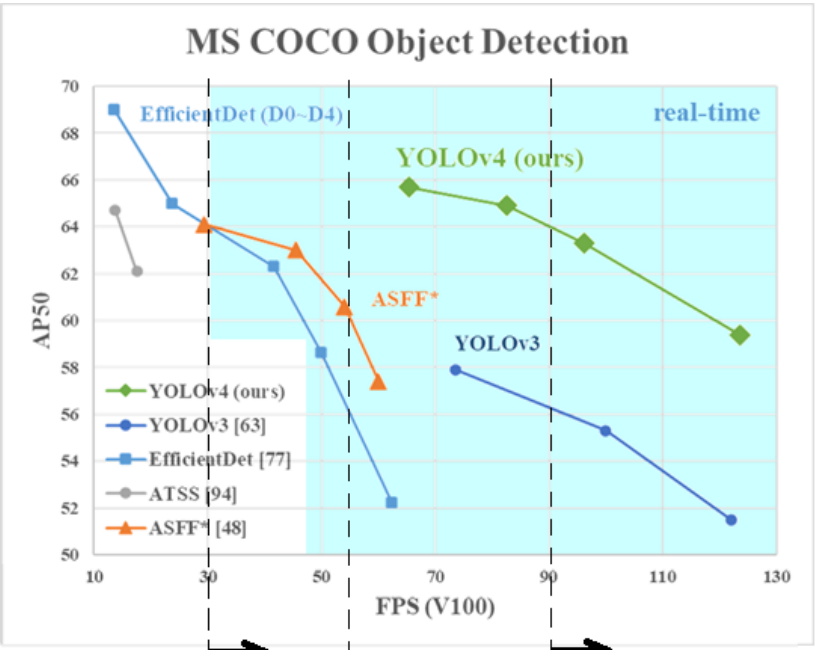
YOLOv4延遲與精確度比較表



資料來源：https://github.com/AlexeyAB/darknet

YOLOv4速度與精確度比較





real-time (30 FPS) on
TitanV/TeslaV100 (server)
250 Watt
$2250 - $8000

real-time (30 FPS) on
RTX 2070 (desktop)
175 Watt
$450

real-time (30 FPS) on
RTX 2060 MXM (Mobile)
80 Watt
$700

9 FPS on
Jetson AGX Xavier (embedded)
30 Watt
$700

real-time (30 FPS) on
TitanV/TeslaV100 (server)
250 Watt
$2250 - $8000

real-time (30 FPS) on
RTX 2070 (desktop)
175 Watt
$450

real-time (30 FPS) on
RTX 2060 MXM (Mobile)
80 Watt
$700

9 FPS on
Jetson AGX Xavier (embedded)
30 Watt
$700

FPS is measured on the Tesla V100 GPU.
For other GPUs with 7.x architecture,
the FPS is estimated based on the TFlops ratio.

資料來源：https://github.com/AlexeyAB/darknet

Yolov4加速工具tkDNN-TensorRT

Nivida GeForce RTX2080 Ti

Network Size	Darknet, FPS (avg)	tkDNN TensorRT FP32, FPS	tkDNN TensorRT FP16, FPS	OpenCV FP16, FPS	tkDNN TensorRT FP16 batch=4, FPS	OpenCV FP16 batch=4, FPS	tkDNN Speedup
320	100	116	**202**	183	423	**430**	**4.3x**
416	82	103	**162**	159	284	**294**	**3.6x**
512	69	91	134	**138**	206	**216**	**3.1x**
608	53	62	103	**115**	150	**150**	**2.8x**
Tiny 416	443	609	**790**	773	**1774**	1353	**3.5x**
Tiny 416 CPU Core i7 7700HQ	3.4	-	-	42	-	39	**12x**

資料來源：https://github.com/AlexeyAB/darknet

預訓練模型

- Yolov3 & Yolov4使用MS COCO(80分類)資料集
- 預訓練權重檔下載
  - yolov4x-mish.weights(381 MB)
  - yolov4-csp.weight(202 MB)
  - yolov4.weights(245 MB)
  - yolov4-tiny.weights(23.1 MB)
  - enetb0-coco_final.weights(18.3 MB)
  - yolov3-openimages.weights(247 MB)

https://github.com/AlexeyAB/Darknet#pre-trained-models

Yolov4-tiny + Webcam測試

- 直接從Github運行Colab範例

https://colab.research.google.com/github/OmniXRI/Yolov4_Colab_User_Datasets/blob/main/Colab_Yolov4_Webcam_Test.ipynb

- 主要步驟：

1. 驗證Nvidia GPU及CUDA版本（可略）
2. 下載DarkNet及Yolov4-tiny預訓練權重檔
3. 修改Make參數
4. 編譯Darknet
5. 測試darknet編譯結果
6. 建立Javascript輔助函式
7. 測試從網路攝影機取得靜態影像並進行物件偵測
8. 建立動態影像處理Javascript函式
9. 從網路攝影機取得動態影像並進行物件偵測

自定義資料集範例程式說明

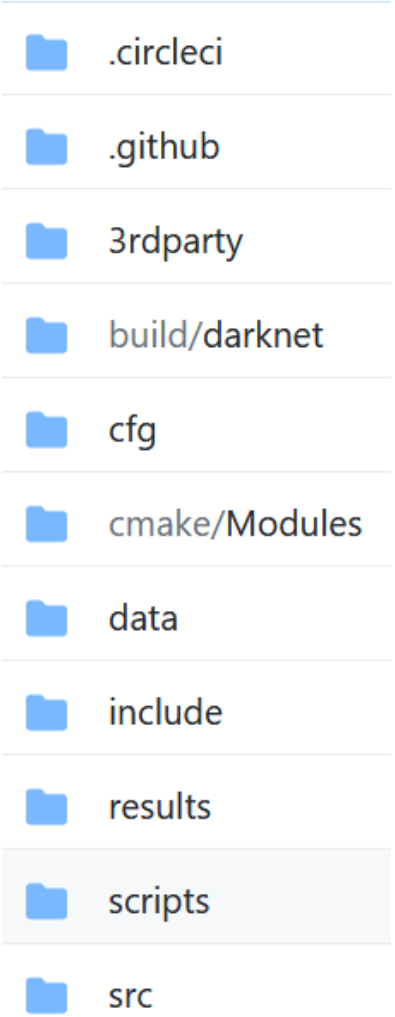
- 從Github下載範例程式（Yolov4 & Yolov4-tiny版）

https://github.com/OmniXRI/Yolov4_Colab_User_Datasets

https://github.com/OmniXRI/Yolov4-tiny_Colab_User_Datasets

- yolov4(_tiny)_training_test.ipynb 為主程式，包含Yolov4(Yolov4_tiny)**預訓練測試**、**自定義資料集訓練及推論測試**。
- my_dataset.zip 包含100張影像及Yolo格式標註檔(*.txt)
- 在個人的Google雲端硬碟新增一個yolov4的路徑，將下載到的所有檔案上傳至雲端硬碟yolov4中。

Darknet檔案夾結構



從Github下載darknet
https://github.com/AlexeyAB/darknet

組態設定
- yolov4.cfg（設定模型組態）
- obj.data（物件資料設定）
- obj.names（物件類別名稱）

資料內容
- *.jpg（測試影像）

修改Makefile並編譯

啟用OpenCV, GPU, CUDNN, CUDNN_HALF

```
%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile

!make # 進行編譯
```

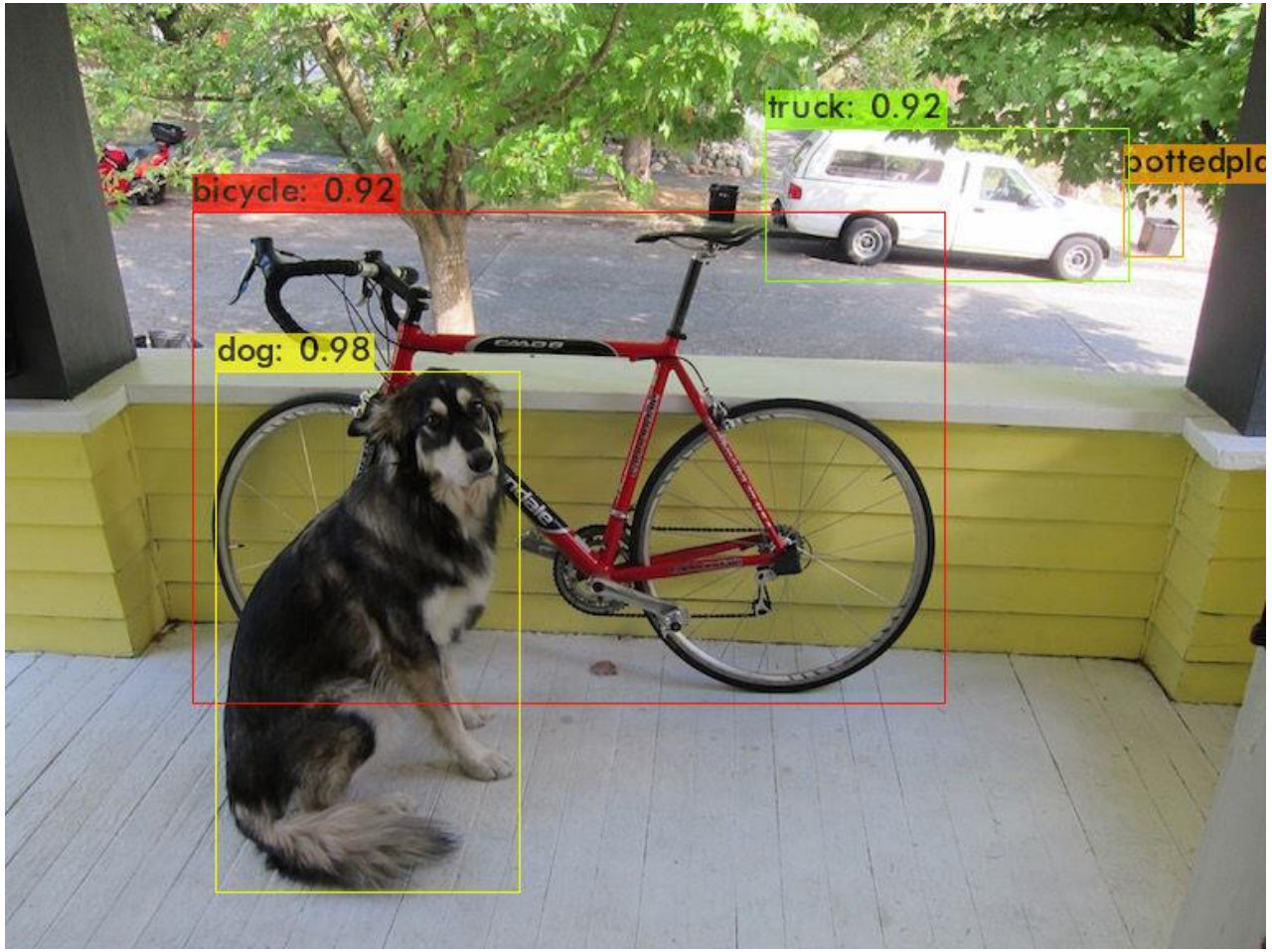
測試Yolov4 (darknet)

```
!./darknet detector test cfg/coco.data cfg/yolov4.cfg /my_drive/yolov4/yolov4.weights  data/dog.jpg
```

```
import cv2 # 導入OpenCV函式庫
from google.colab.patches import cv2_imshow # 導入Colab.patches函式庫
```

```
imgResult = cv2.imread('predictions.jpg') # 讀入結果影像
cv2_imshow(imgResult) # 顯示結果影像
```

預訓練模型測試結果



data/dog.jpg: Predicted in 165.845000 milli-seconds.

bicycle: 92%
dog: 98%
truck: 92%
pottedplant: 33%

自定義模型訓練

1. 準備資料集（影像檔）
2. 使用LabelImg標注工具製作符合YOLO格式
3. 將步驟3準備好的相關設定及預訓練檔從 Google Drive /yolov4 路徑下複製到Colab指定路徑下，包含下列六種檔案。
   2.1 obj.data （物件資料設定）
   2.2 obj.names （物件類別名稱）
   2.3 yolov4.cfg （設定模型組態）
   2.4 train.txt （訓練內容，另含原始影像壓縮檔）
   2.5 valid.txt （驗證內容，另含原始影像壓縮檔）
   2.6 pre-trained.weight （預訓練權重檔 yolov4.conv.137(yolov4), yolov4_conv.29(yolov4-tiny)）

https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137

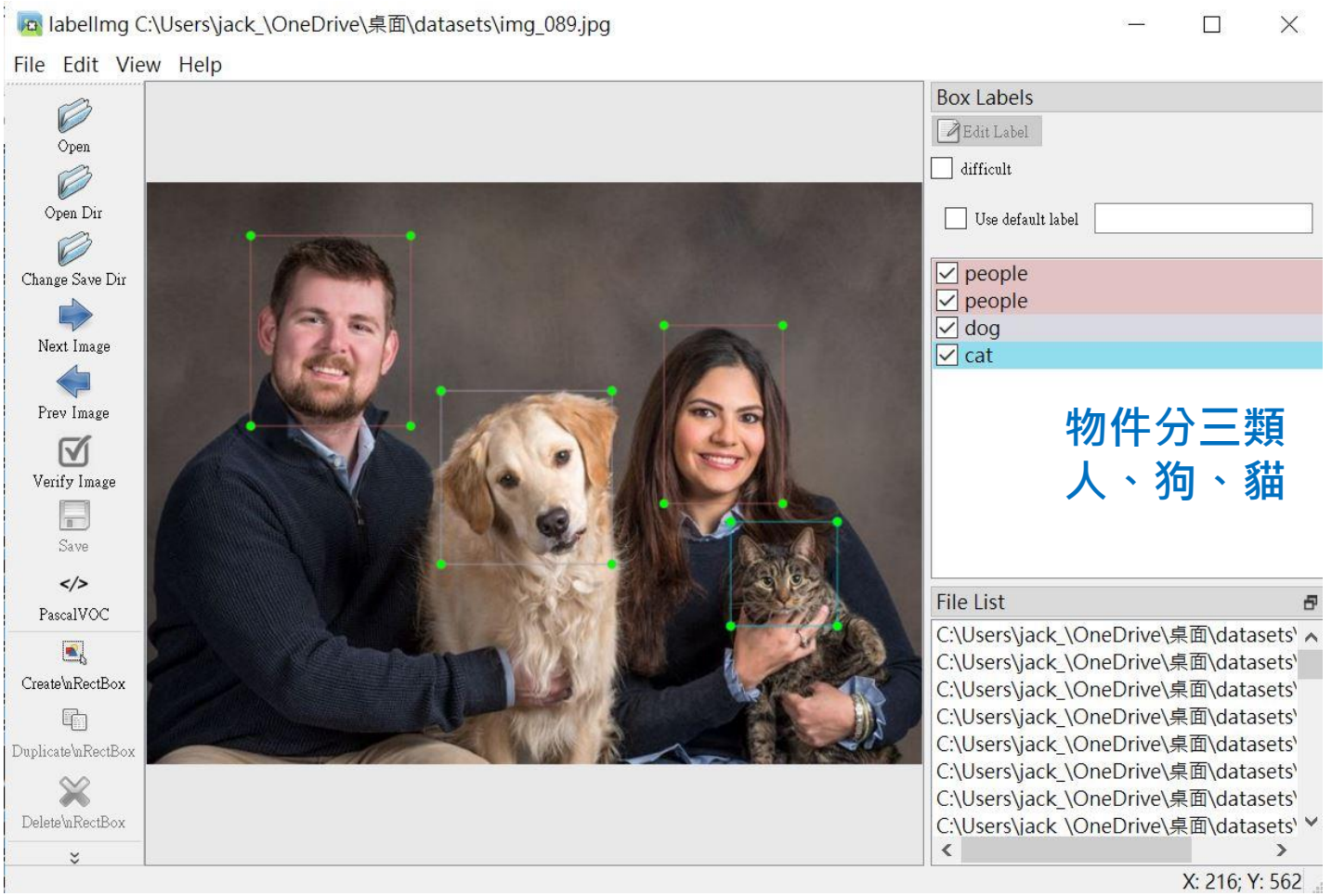
自訂義資料集

100張影像，其中img_001-img_080作為訓練用，其餘20張作為驗證用。

LabelImg物件標註

預設標註檔為VOC格式，YOLO格式要用另存新檔。

選擇輸出格式VOC(*.xml) / YOLO(*.txt)



https://github.com/tzutalin/labelImg

建立自定義訓練相關檔案

- my_obj.data

classes = 3 **(類別數量）**
train = data/my_train.txt **（訓練集）**
valid = data/my_test.txt **（驗證集）**
names = data/my_obj.names **（類別名稱）**
backup = /my_drive/yolov4/ **（自動備份路徑）**

- my_obj.names

dog
cat
people

修改自訂義組態檔

- my_yolov4_custom.cfg

  (從/cfg/yolov4_custom.cfg複製而得)

```
batch=64 # line 6
subdivisions=16 # line 7
width=416   # line 8, 32倍數
height=416  # line 9, 32倍數

max_batches = 6000 # line 20, classes*2000

steps=4800,5400  # line 22, max 80%, 90%
```

有三處要修改
filter數量為(classes+5)*3

```
[convolutional]
filters=24 # 第963, 1051, 1139列

[yolo]
classes=3 # 第970, 1058, 1146
```

執行自定義模型訓練

- 正常（重頭）執行

```
!./darknet detector train data/my_obj.data cfg/my_yolov4_custom.cfg yolov4.conv.137 -dont_show
```

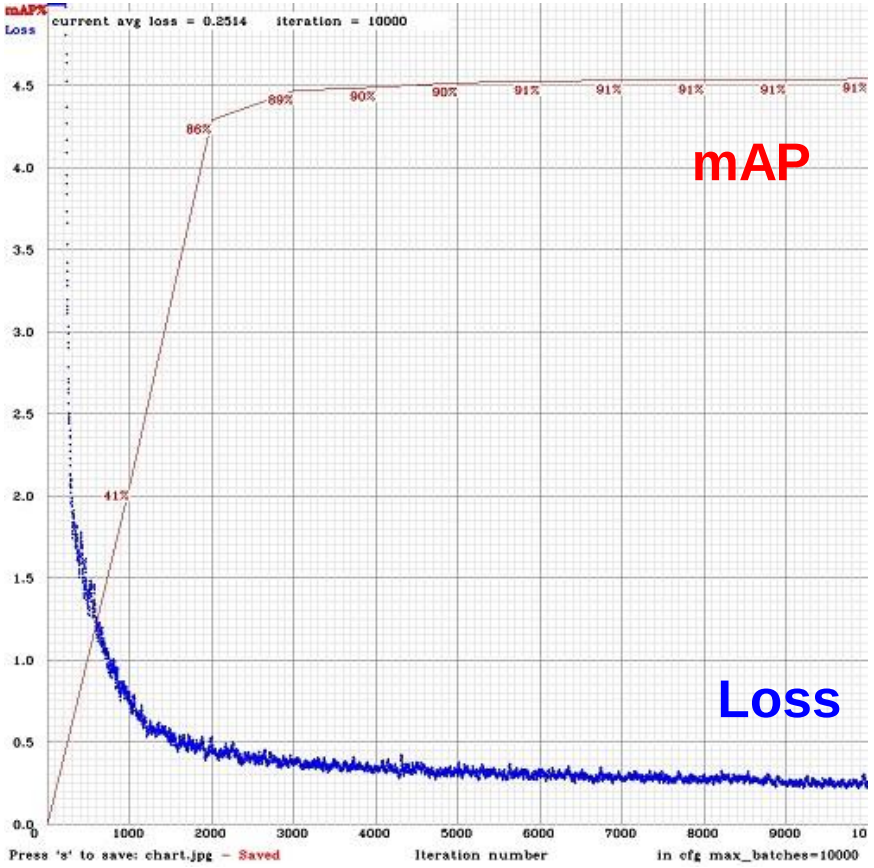
每1000次會產生一個暫存檔my_yolov4_custom_x000.weights ，最後會產生一個完成檔my_yolov4_custom_final.weights

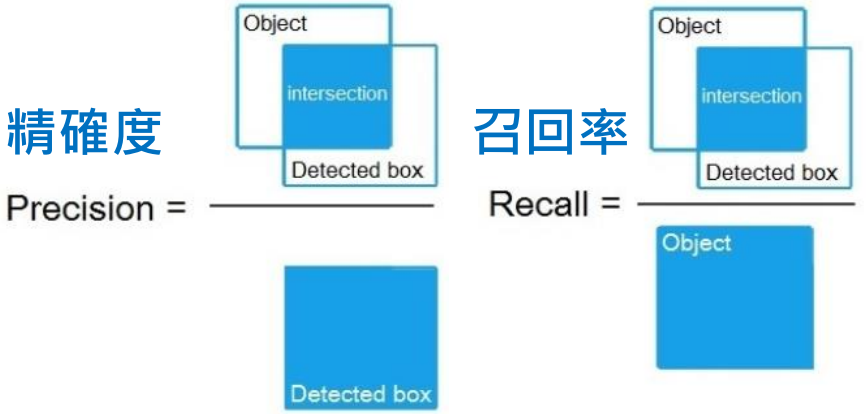
- 接續執行

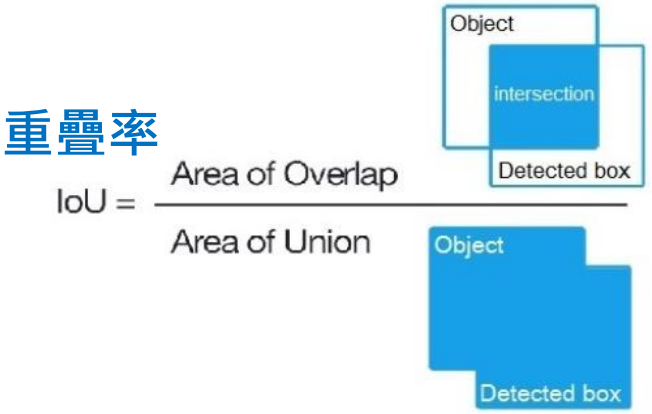
```
!cp /my_drive/yolov4/my_yolov4_custom_last.weights backup/

!ls backup/

!./darknet detector train data/my_obj.data cfg/my_yolov4_custom.cfg backup/my_yolov4_custom_last.weights -dont_show
```

訓練損失Loss與平均精確度mAP

./darknet detector train data/obj.data yolo-obj.cfg yolov4.conv.137 -map





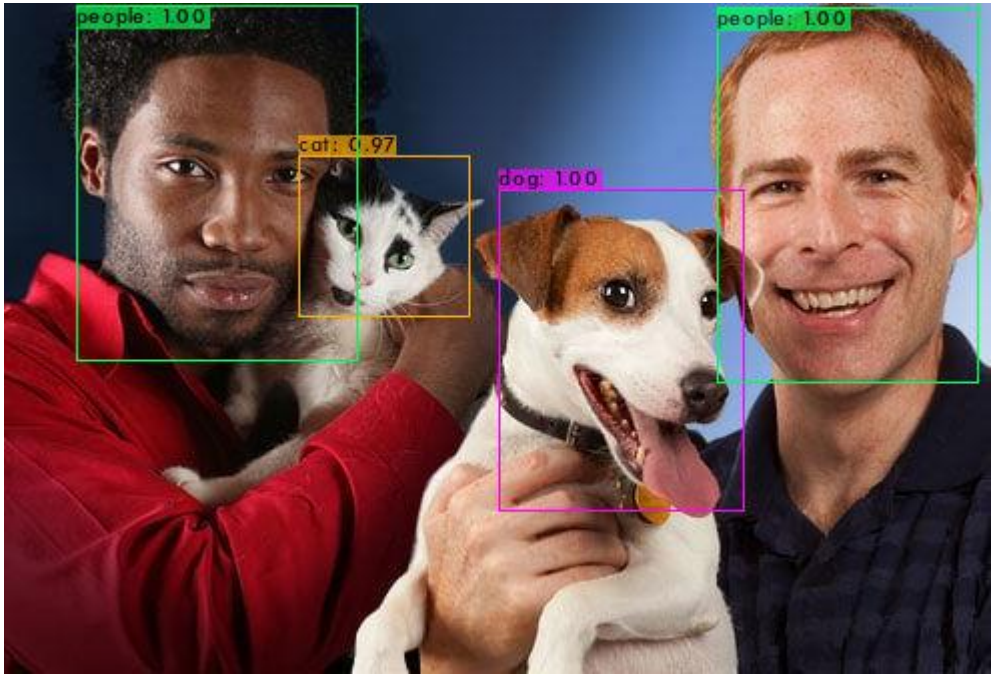


資料來源：https://github.com/AlexeyAB/darknet

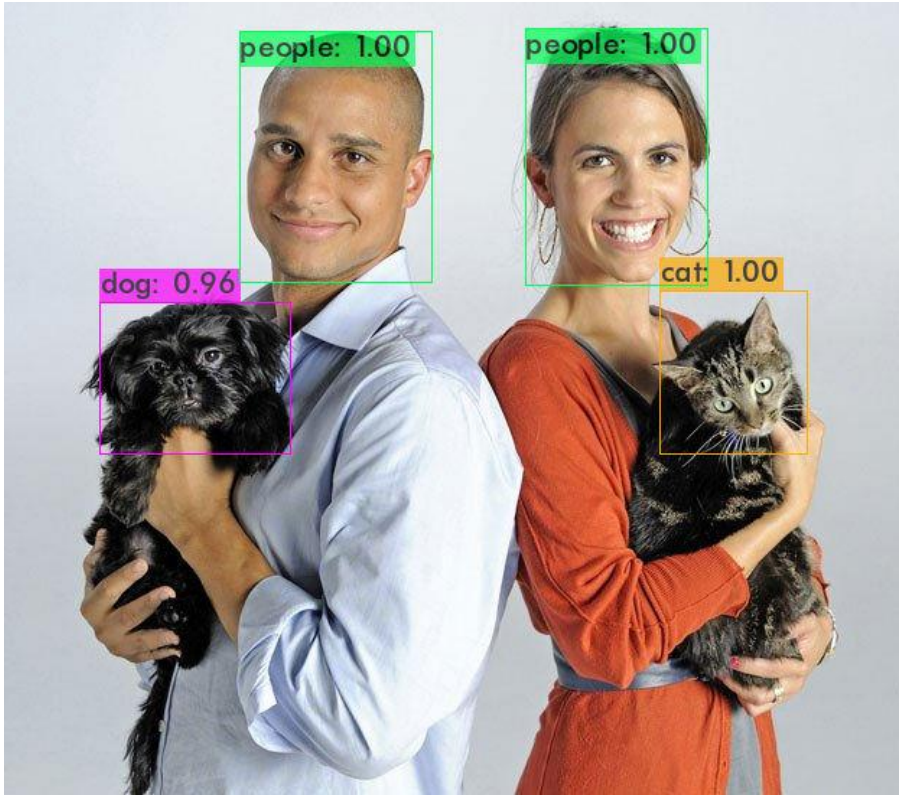
執行自定義模型推論

```
!./darknet detector test data/my_obj.data cfg/my_yolov4_custom.cfg backup/my_yolov4_custom_final.weights /my_drive/yolov4/test01.jpg


import cv2 # 導入OpenCV函式庫

from google.colab.patches import cv2_imshow # 導入Colab.patches函式庫


imgResult = cv2.imread('predictions.jpg') # 讀入結果影像

cv2_imshow(imgResult) # 顯示結果影像
```

自定義模型測試結果



test01.jpg



test02.jpg

Yolov4 vs. Yolov4-tiny

	Yolov4	Yolov4-tiny
預訓練 Config	yolov4.cfg	yolov4-tiny.cfg
預訓練 Weight	yolov4.weights (245 MB)	yolov4-tiny.weight (23.1 MB)
自定義 Config	yolov4-custom.cfg	yolov4-tiny-custom.cfg
自定義 Weight	yolov4.conv.137 (162 MB)	yolov4-tiny.conv.29 (19 MB)
適用領域	雲端 / 桌機	單板電腦 / 小型AI晶片

參考資料：https://github.com/AlexeyAB/darknet#how-to-train-to-detect-your-custom-objects

參考文獻

- pyimagesearch - Introduction to the YOLO Family

https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/

- YOLO Github

https://pjreddie.com/darknet/yolo/

- Darknet Github

https://github.com/AlexeyAB/darknet

- 完整範例程式

https://github.com/OmniXRI/Yolov4_Colab_User_Datasets (yolov4)

https://github.com/OmniXRI/Yolov4-tiny_Colab_User_Datasets (yolov4-tiny)

- 許哲豪，如何以Google Colab及Yolov4-tiny來訓練自定義資料集─以狗臉、貓臉、人臉偵測為例

https://omnixri.blogspot.com/2021/05/google-colabyolov4-tiny.html